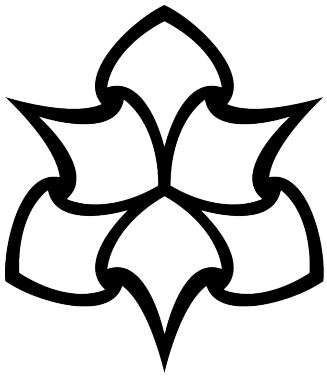


Oswaldo Terán, Bruce Edmonds and Steve Wallis

Centre for Policy Modelling
The Business School

Enveloping Tendencies in Fragments of a Simulation Theory

WP00/05

December 2000
ISSN 1471-857X

Oswaldo Terán
Bruce Edmonds
Steve Wallis

Centre for Policy Modelling
The Business School
Manchester Metropolitan University
Aytoun Building,
Aytoun Street,
Manchester M1 3GH

Tel: **0161 247 6478**

E-mail: o.teran@mmu.ac.uk

b.edmonds@mmu.ac.uk

s.wallis@mmu.ac.uk

URL http://www.cpm.mmu.ac.uk/

Biography

Oswaldo Terán is a member of the Centro de Simulación y Modelos (CESIMO: Centre for Simulation and Modelling) and the Department of Operations Research of the University of Los Andes, Venezuela (http://cesimo.ing.ula.ve/).

Bruce Edmonds is a Senior Research Fellow in Logic and Formal Methods at the Centre for Policy Modelling (http://www.cpm.mmu.ac.uk/~bruce/).

Steve Wallis is a Senior Research Fellow in Computer Science at the Centre for Policy Modelling (http://www.cpm.mmu.ac.uk/~stevew/).

Abstract

In this paper we present a methodology aimed at systematically exploring the 'envelope' of simulation trajectories allowing us to prove the necessity of tendencies respect to Fragments of a Simulation Theory. More well-grounded conclusions about tendencies in a simulation can be dig up than those given by existing methods like Monte Carlo techniques and Scenario Analysis where partial investigation of trajectories are performed -this is helpful in research areas such as Social Simulation, Management and Policy Analysis. We propose a method for searching for tendencies and proving their necessity, in Multi Agent Systems, relative to a range of parameterisations of the model and agents' choices, and to the logic of the simulation language. Additionally, a computational procedure that helps implement this exploration by translating the Multi Agent Systems simulation into a constraint-based search over possible trajectories by 'compiling' the simulation rules into a more specific form is proposed and exemplified.

Key words: Social Simulation, Policy Analysis, Multi-agent Systems, Model, Proof, Emergence, Tendencies

Behaviour in a Simulation

Behaviour observed in a simulation can be classified in three groups in accordance to two criteria. The criteria are: first, whether the observed behaviour is or is not given in the simulation design; and second, for behaviour not given in the simulation design, whether it is or it is not associated with aspects well understood in the target system.

That behaviour given in the design is useful for verifying the simulation and is not of interest for understanding the target system or the simulation itself. Second, behaviour not given in the simulation design and well understood in the target system might be used for validating the simulation, but it will not be valuable for understanding the target system. And finally, we have those aspects of simulation related to behaviour little understood in the target system and obviously not given in the simulation design. We will call tendencies not given in the simulation and difficult to understand 'emergent tendencies' (for a related notion of emergence of tendencies, see e.g., Edmund et al. 1999).

A simulation hopefully will inform about this last sort of behaviour. In fact, the need to understand better certain kind of behaviour in a target system is what motivates a simulation in many areas of research. In areas such as social simulation, it is of particular interest to analyse processes and to understand better tendencies in social behaviour (Carley et al., 1998). In management and policy analysis simulation is valuable to guide and inform managers and policy analysts, assisting them for taking decisions (Wack, 1985a and 1985b; Domingo et al., 1996). Well-grounded information will help in all these areas of research to test theories and hypothesise about the simulation and the target system, and will assist more convincingly managers and policy analysts.

It is of particular interest for modellers in, for example, the named areas of research, to analyse the commonality of emergent tendencies in different simulation trajectories as this allows them to draw conclusions about the theory implied in the simulation. However, usually there is a trade-off between the richness of the study in terms of the number of explored trajectories (sometimes related to how fine-grained

the model is) and the amount of required computational resources. The finer the model the more "realistic" the simulation model will be, but also the more intricate the analysis of the simulation will be.

A typical case where this analysis is crucial is in Multi-Agent Based Simulation of social systems. There, modellers may attempt to generate in the lab certain "complex" behaviours in a whole population as the result of the interaction of simpler. Unforeseen behaviour of individuals and unpredictable tendencies in the behaviour of the whole population can arise (Edmonds, 1999).

The lack of alternative methodologies and tools for appropriate exploration and analysis of the dynamics of a simulation are presently a factor, which limits the comprehension of emergent tendencies. Present methods include examining individual trajectories as in Scenario Analysis (Domingo et al., 1996) and statistical sampling as in Monte Carlo techniques (Zeigler, 1976). They consist in partial explorations of simulation trajectories. In the first approach the partialness rests in a criterion chosen by the modeller, and in the second method trajectories are picked up randomly. In both of them the scope of conclusions is limited as they are incompletely grounded.

It is our purpose in this paper to complement those methods with an alternative way of exploring and analysing the simulation by systematically and automatically enveloping all possible trajectories in a substantial fragment of a simulation theory. More specifically, this paper proposes a complete search of trajectories for a range of parameterisations and agents' choices. This kind of search corresponds to a model exploration in Theorem Proving (Bonacina, 1998). Consequently, conclusions will be more well-grounded and can be applied in wider theory than when using the named alternative methods.

Enveloping Tendencies in Simulation Trajectories: a Constrained Search over Possible Models

1.1 Constrained Exploration of Trajectories

We propose the use of an exhaustive constraint-based search over a range of possible trajectories in order to establish the necessity of postulated emergent tendencies. Thus a subset of the possible simulation parameterisations and agent choices are specified; the target emergent tendencies are specified in the form of negative constraints; and an automatic search over the possible trajectories performed. The tendencies are shown to be necessary with respect to the range of parameterisations and non-deterministic choices by first finding a possible trajectory without the negative constraint to show the rules are consistent and then showing that all possible trajectories violate the negation of the hypothetical tendency when this is added as a further constraint (See figure 1). This corresponds to a Model Based exploration in Theorem Proving (Bonacina, 1998).

1.2 Proving the Necessity of a Tendency

We want to be able to generalise about tendencies going from observation of individual trajectories to observation of a group of trajectories generated for certain parameters and choices. Actually, we want to know if a particular tendency is a necessary consequence of the system or a contingent one. For doing this we propose to translate the original Multi Agent System along with the range of parameterisations and agents' choices into a platform (described in the next section) where the alternative trajectories can be unfolded. Each trajectory will correspond to a possible trajectory in the original Multi Agent System. Once one trajectory is shown to satisfy the postulated tendency another set of parameters and agents' choices is selected and the new trajectory is similarly checked. If all possible trajectories are successfully tested, the tendency is proved to be necessary relative to the logic of the simulation language, the range of parameterisations and agents' choices.

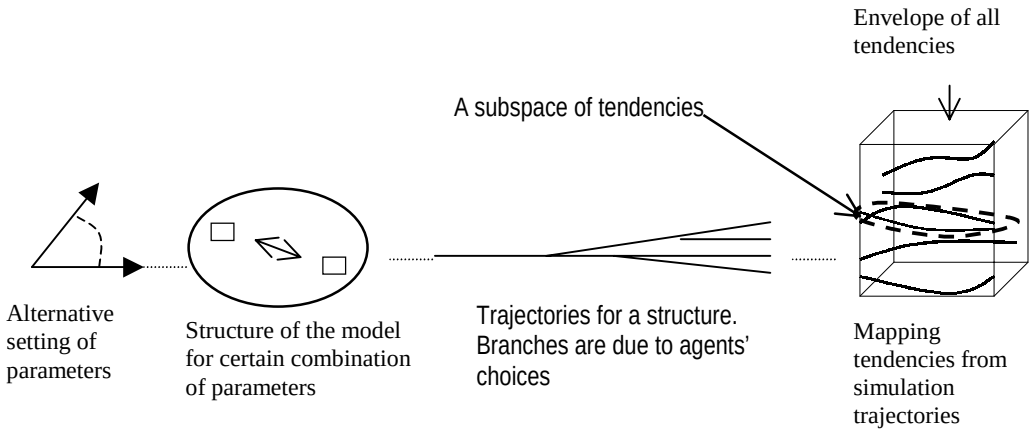


Figure 1. A constraint-based exploration of possible simulation

The idea is to translate the Multi Agent System into a constraint-based platform in an automatic or near automatic way without changing the meaning of the rules that make it up in order to perform this automatic testing. In this way a user can program the system using the agent-based paradigm with all its advantages; inspect single runs of the system to gain an intuitive understanding of the system and then check the generality of this understanding for fragments of the system via this translation into a constraint-based architecture.

In the example shown below, all trajectories are explored for one combination of parameters, eight agents' choices per iteration and seven iterations. A simple tendency was observed characterised by a mathematical description of its boundaries. This characterisation was handled as a theorem. The theorem was proved to be necessary following a procedure similar to the one described in the previous paragraph.

1.3 What is New in This Model-Constrained Methodological Approach

It is our goal in this paper to propose an alternative approach for exploring and analysing simulation trajectories. It will allow the entire exploration and subsequent analysis of a subspace of the whole space of simulation trajectories. We are suggesting the generation of trajectories in a semantically constrained way. Constrictions will be context-dependent (over the semantics of the trajectory itself) and will be driven via the introduction of a controller or meta-module.

Like Scenario Analysis, the idea is to generate individual trajectories for different parameterisations and agents' choices but unlike Scenario Analysis the exploration is constrained to only certain range of parameters and choices.

Akin to Monte Carlo techniques it explores only part of the total range of possible trajectories. But, unlike Monte Carlo studies it explores an entire subspace of (rather than some randomly generated sample) trajectories and is able to give definitive answers for inquires related to the dynamics of the simulation in that subspace.

Towards the Implementation of a Suitable Platform for the Envelope of Trajectories using Strictly Declarative Modelling Language

Strictly Declarative Modelling Language (Moss et al., 1998) is the declarative language where we have built the Multi-Agent System in which the experiments have been developed. As a source of comparisons and ideas, the model has also been programmed in a Theorem Prover (Chiang et al., 1973; McCune, 1995; Wos, 1988).

A Theorem Prover is a computational system aimed at exploring the theory embedded in a set of clauses and a set of inference rules, in a search for a proof of a given theorem in such a theory. Theorem prover systems have been developed with different purposes than logic programming languages like Prolog, but there exist theorem provers written as extensions of these systems (e.g., as extensions of Prolog). Theorem provers have become popular, for example, for proving mathematical theorems and for verifying computational programs (Wos, 1988). Nevertheless, the idea of proving theorems in a simulation theory not given in the simulation design and even no well understood by the modeller (like emergent tendencies) is a novel idea coming from the necessity of understanding better processes in simulations of complex systems and, particularly, in simulations of social systems. However theorem provers are more oriented for doing symbol manipulation and for proving in formal logic than for simulation and numerical manipulation, they can provide valuable ideas for exploring theorems in a simulation theory. These ideas hopefully will be helpful for developing methodologies and techniques for proving in simulation of complex systems, which can be more comfortably implemented in simulation languages such as Strictly Declarative

Simulation Language. In order to make possible such implementations, certain characteristics and features will be required in these simulation languages. For example well-grounded underlying logical properties and appropriate mechanisms for exploring the simulation theory.

Strictly Declarative Modelling Language offers desirable features for simulation experiments as compared to imperative programming. For the social simulation community those features seem to be of particular interest when facilitating the exploring and analysis of the dynamics of the simulation (Moss et al., 1997).

Among the good features Strictly Declarative Modelling Language (SDML) offers for a model-based exploration of simulation trajectories, we have:

- Good underlying logical properties of the system. Good underlying logical properties in the sense of well grounded. SDML's underlying logic corresponds to a fragment of the Strongly Grounded Autoepistemic Logic (SGAL) described by Kurt Konolige (Konolige, 1995).

- Its backtracking mechanism facilitates the exploration of alternative trajectories via the splitting of simulation paths according to agent's choices and model's parameters.

- Efficient forward chaining assumptions manager in SDML tracks the use of assumptions. Assumptions result from choices.

- A collection of useful primitives relevant to social simulation.

- Meta-agent for automatic translation of rules. A meta-agent (meta, for our purposes) is an agent "attached" to another agent as a controller; it is able to write rules in that agent. This is used here not as an agent per se but as a module used to 'compile' rules into an efficient form as well as to monitor and control the overall search process and goals.

- A mechanism for an automatic and static analysis of rule dependencies.

- Simple negative contradiction generation via false predicate: P => •

- User defined backward chaining clauses useful to be used as demodulators.

Implementing a Suitable Constraint-Based Programming Platform

The main goal of the programming strategy to be described is to increase the efficiency in terms of simulation time, thus making an efficient constraint-based search possible. The improvements will be achieved by making the rules and states more context-specific. This enables the language's inference engine to exploit more information about the logical dependencies between rules and thus increase the efficiency. Thus this can be seen as a sort of 'practical compilation' process, which undoes the agent encapsulation in order to allow the more efficient exploration of its behaviour. In particular we split the transition rules into one per simulation period, and also by the initial parameters. This necessitates a dynamic way of building rules. This is done via a controller, which generates the rules at the beginning of the simulation.

1.4 An Overview of the System

We implemented the proposed architecture in three modules; let us call them *model*, *prover* and *meta*. The following diagram illustrates this:

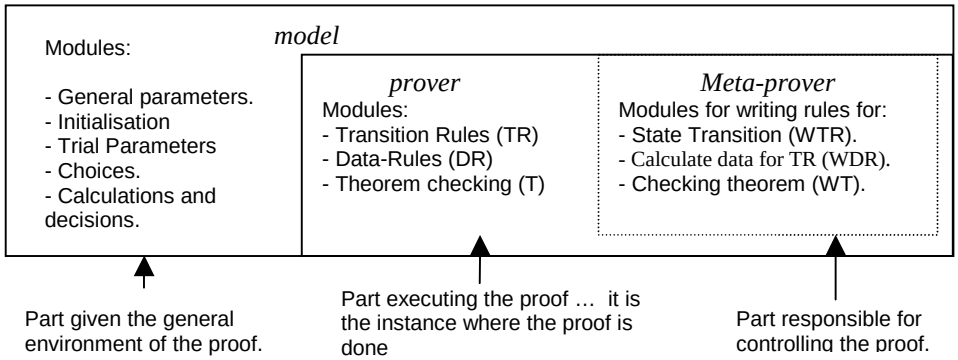


Figure 2. Illustration of the system's parts.

1.5 Description of System Modules

We have found it convenient to distinguish and model as distinct entities three basic elements of a simulation: the static structure of the model, the dynamics of the simulation and the way this dynamics is "managed" by certain meta-rules or by a controller. Each of those entities is programmed in a different module:

- model, sets up the structure of the model, that is, it gives the environment of the simulation: range of parameters, initialisations, alternative choices and basic (backward chaining) rules for calculations.

- prover, generates the dynamics of the simulation. This is a sub-module of model (i.e. it is contained in model). This will basically contain the transition rules, auxiliary rules for generating pre-processing required data and the conditions to test the necessity of the theorem. All of them are rules to be executed while the simulation is going on.

- meta, is responsible for controlling the dynamics of the simulation. Its meta-rules write the transition rules and the theorem in (as well as others required by) the module prover. A picture of the system is given in Figure 2.

1.6 Program Dynamics

Modules' rules are executed in the following sequence:

- model: initialising the environment for the proof (setting parameters, etc..)

- meta: creating and placing the transition rules in prover.

- prover: carrying on the simulation using the transition rules and backtracking while a contradiction is not found.

The program backtracks from a path once the conditions for the theorem are verified, then a new path with different choices and/or parameters is picked up.

1.7 Split of the Rules: a Source of Efficiency

In forward chaining simulation the antecedent retrieves instance data from the past in order to generate data for the present (and maybe the future):

past facts → present and future facts

Traditionally, the set of transition rules are implemented to be general for the whole simulation. A unique set of transition rules is used at any simulation iteration.

As the simulation evolves, the size of the database increases and the antecedents have to discriminate among a growing amount of data. At iteration-i , there would be data from (i-1) alternative days matching the antecedent. As the simulation evolves it becomes slower because of the discrimination the program has to carry out among this (linearly) growing amount of data.

Using the proposed technique, we would write a transition rule for each simulation time (see figure 3). The specific data in the antecedent as well as in the consequent could be instanced. Where possible, a rule for each datum, the original rule will generate, would be written. The splitting of rules lets us discriminate among the transition rules for different simulation times given a more specific instancing of data.

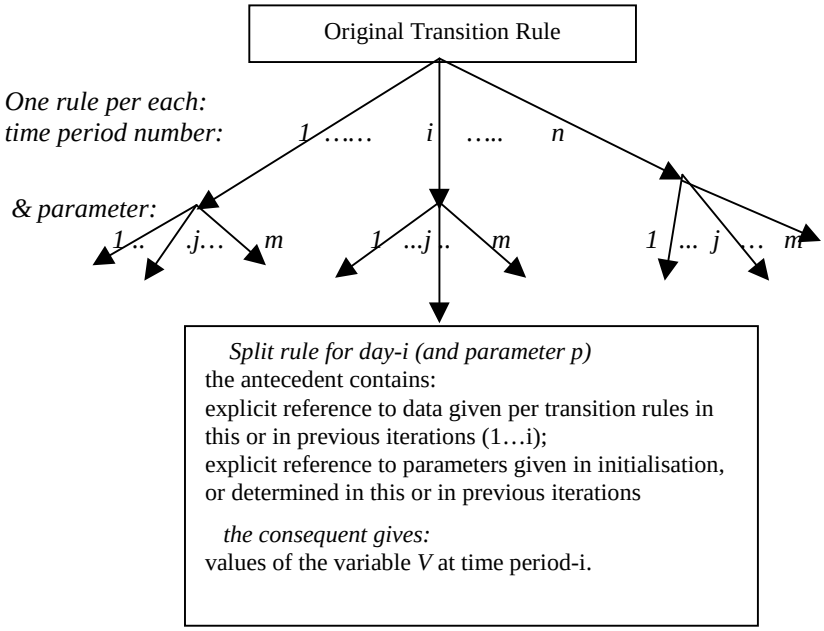


Figure 3. Splitting of rules by time period and a combination of

1.8 Measuring the Efficiency of the Technique

Comparing the two programs, the original Multi Agent System simulation and the constraint-based translation we obtain a speed up by a factor of O(NM), where N is the average number of agents instantiated by a rule and M is the number of iterations. SDML already has facilities for discriminating among iterations, but their use is not convenient for the sort of simulation we are doing (exploring scenarios and/or proving) because of the difficulties for accessing data from any time step at any time. If we had used this facility still the simulation would have been speeded up by N. Notice that all these values are only estimations because a program stops trying to fire a rule as soon as it finds out that one of its clauses is false.

It is clear that the greater the number of entities in the simulation or the number of iterations, the larger the benefits from the technique. We must notice that the speeding up of the simulation is only one dimension of the efficiency given by the technique.

An Example

A simple trader-distributor model was built in SDML. It resembles basic characteristics that can be observed in many empirical models but it is ideal in the

sense that it is not a representation of any particular empirical model. There are six agents: three Distributors and three Traders (see left side of figure 4).

This model was rebuild first in the Theorem Prover OTTER (McCune, 1995), one of the more successful theorem provers, in order to find ideas for a more efficient implementation than a traditional Multi Agent System simulation and then also in SDML using the proposed modelling strategy. In the new SDML model the exploration of possibilities was speeded up by a factor of 14. Also, the model built in OTTER, though faster than the original model in SDML, was several times slower than the improved SDML model.

Translating the Multi Agent System model into the Constraint Based architecture:

The idea is to build a new model in SDML facilitating reasoning about the whole simulation by having a single rulebase-database were dependencies previously hidden in the hierarchies of agents, time levels and modules are revealed and

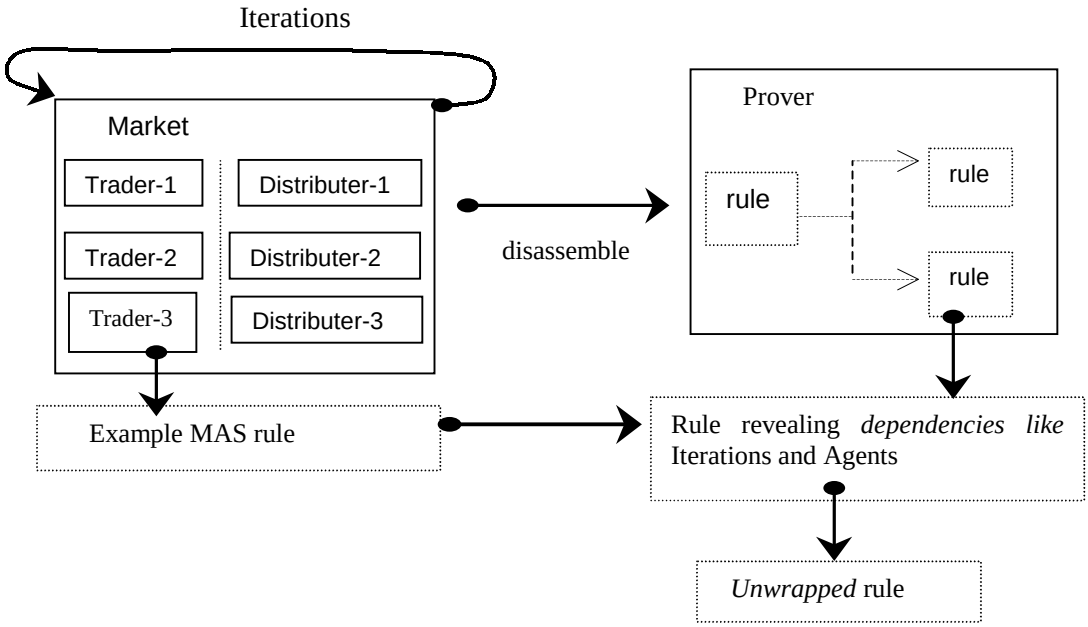


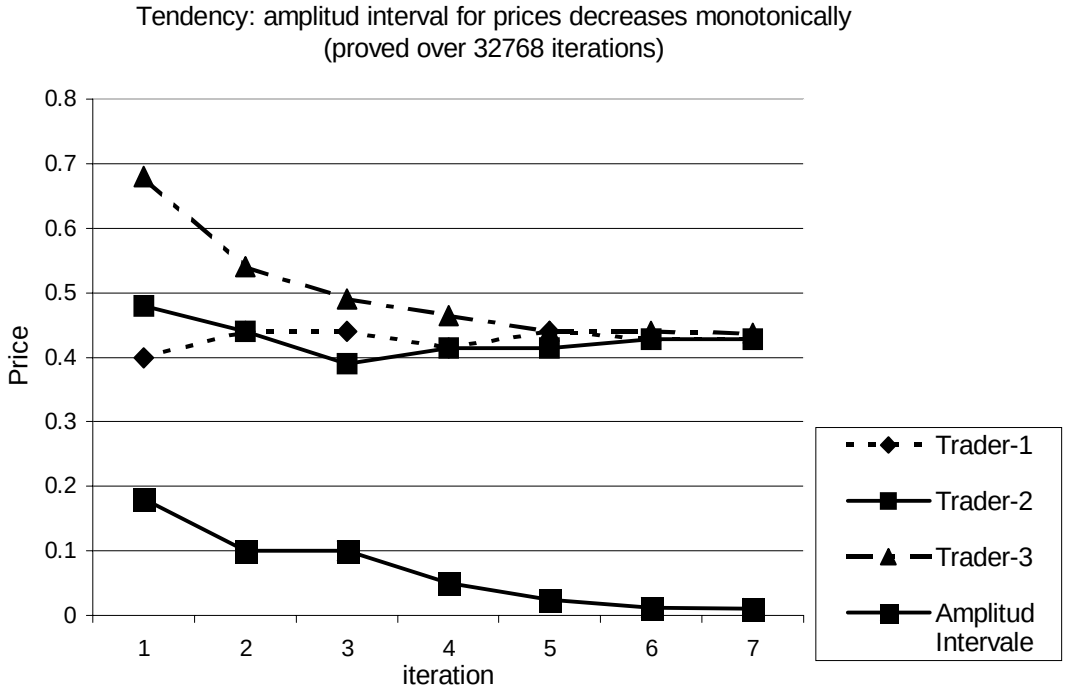
Figure 4. Translating the original MAS into the Constraint Based architecture: revealing dependencies and unwrapping rules

exploited to 'unwrap' the rules and speed up the simulation (see right side of figure 4).

1.9 What the Technique Enables

In this example, the described technique was used to prove that the size of the interval of prices (that is: biggest price - smaller price, each day) decreases over time during the first six iterations over a range of one parameterisation and eight choices for the agents at each iteration. An exponential decrease of this interval was demonstrated in all the simulation paths. A total of 32768 simulation trajectories were tested. It was not possible to simulate beyond this number of days because of the limitations imposed by computer memory. The complete search process took only 24 hours.

The tendency is expressed as an envelope rather than as a central tendency, as e.g., in Monte Carlo techniques. At certain iteration in a trajectory, the bounds of the tendency are given by the highest price and lowest prices among all Traders' prices (i.e. [highest Price, lowest Price]). Its bounds for certain iteration for a set of explored trajectories are given as the union of the intervals got for that iteration in all



Graph 1. Tendency observed in a trajectory

explored trajectories (see graph 1). So, the idea is to envelope the tendency no for only one trajectory but for a subset of trajectories, those given somehow as a

subspace of the whole set of possible simulation trajectories. This allows managers, policy analysts and social modellers to get better informed than when using central measures of tendencies.

Though the tendency we have shown is simple and quantitative, the technique is applicable in more interesting cases of emergent tendencies, even if they have a qualitative nature.

Other Approaches

In OTTER (and similar Theorem Provers) the set of simulation rules and facts (atoms) is divided into two sets (this strategy is called support strategy) (McCune, 1995):

One set with “support” and the other without it. The first one is placed in a list called “SOS” and, the second one, in the list “USABLE”. Data in USABLE is “ungrounded” in the sense that the rules would not fire unless at least one of the antecedents is taken from the SOS list. Data inferred using the rules in USABLE are placed in SOS when they are not redundant with the information previously contained in this list, and then used for generating new inferences. The criteria for efficiency are basically subsumption and weighting of clauses.

Rules are usually fired in forward chaining but backward chaining rules and numerical manipulations are allowed in the constructs called “demodulators” (Wos, 1988).

In simulation strategies like event-driven simulation or partition of the space of rules, in declarative simulation, are used. The criteria for firing rules is well understood, and procedures like weighting and subsumption usually are not necessary. Additionally, redundant data could be avoided in Multi Agent System with a careful programming.

The advantages given for the weighting procedure in OTTER are yielded in Multi Agent System systems like SDML by procedures such as partitioning, where chaining of the rules allows firing the rules in an efficient order according to their dependencies.

Among other approaches for the practical proof of Multi Agent System properties, the more pertinent might be the case conducted by people working in DESIRE (Engelfriet et al., 1998). Engelfriet et al. propose the hierarchical verification of Multi Agent System properties, and succeeded in doing this for a system.

However, their aim is limited to verification of the computational program – it is proved that the program behaves in the intended way. It does not include the more difficult task, which we try to address, of establishing general facts about the dynamics of a system when run or comparing them to the behaviour observed in other systems (Axtell et al., 1996).

Koen et al. (2000) use contextual information for adding flexibility in behaviour of agents' using preference models. In particular they propose building agents able to 'adapt' their plans in an environment with uncertainty and soft deadlines by using a context-sensitive planning. He claims these agents have more 'realistic' preference models than those commonly used in other approaches. Their idea of a context sensitive planning is comparable to our idea of context driven exploration of simulation trajectories proposed in the second level of architectural transformations.

The Riley et al. (2000)' paper is related with understanding Multi Agent System and observing aspects of their dynamics -in this sense related to the work presented in this article. Concretely, they propose a 'layered disclosure by which autonomous agents have included in their architecture the foundations necessary to allow them to display upon request the specific reasons for their actions'. In fact, this mechanism permits a modeller to check the state of the internal model of an agent at certain simulation time. This sort of mechanism is programmable in SDML by writing the specific rules for required reports, or, stopping the simulation and then writing the consulting rules in the 'Experiment tag' of the appropriated agent. In addition, SDML allows us to return to previous states in the simulation. The analysis of the dynamics of a simulation they propose is quite simple and not so useful for understanding aspects of the simulation related with the theory implicit in the simulation. They do not address the more fundamental aspect of analysing tendencies (regularities over time) but rather aspects at certain isolated simulation instants.

Conclusions and Further Work

We have proposed a method for a Model-Based proof of emergent tendencies in fragments of a simulation theory. In particular we have suggested a constraint-based semantically oriented exploration of all simulation trajectories following a forward chaining inference procedure. Once a tendency is identified the idea is to prove its necessity relative to the logic of the simulation language, a range of parameterisations and agents' choices. The proof will be relative to the fragment of the theory defined by such a range of model's parameters, agents' choices and the logic of the program.

A platform to implement this methodology has been proposed. It consists of a modular structure according to strategic parts of a simulation: a first module, model, sets up the static structure of the simulation; then a second module, prover, generates the dynamics of the simulation; and finally a meta-module is responsible for controlling the dynamics of the simulation. The second characteristic of this platform is a partitioning of the space of rules and splitting of transition rules by STI, parameters and choices.

The suggested method allows a modeller to draw more well-grounded conclusions than when using existing methods such as scenario analysis and Monte Carlo techniques. It is valuable in applications in soft systems such as management, policy analysis and social simulation. It assists more convincingly managers and permits researchers in these areas to test theories and to elaborate more well-grounded hypothesis about the behaviour of both the simulation and the target system.

Further work should be orientated to develop even more efficient (in terms of the relation explored simulation space - required computational resources) architectures for investigating and proving behaviour in simulations. We believe Constraint Logic Programming, and in particular Rule Based Constraint Logic Programming, is a promising source of ideas (Frühwirth, 1994; Frühwirth et al., 1992).

Acknowledgements

SDML has been developed in VisualWorks 2.5.2, the Smalltalk-80 environment produced by ObjectShare. Free distribution of SDML for use in academic research is made possible by the sponsorship of ObjectShare (UK) Ltd. The research reported here was funded by CONICIT (the Venezuelan Governmental Organisation for promoting Science), by the University of Los Andes, and by the Faculty of Management and Business, Manchester Metropolitan University.

References

1. Axtell, R., R. Axelrod, J. M. Epstein, and M. D. Cohen (1996), Aligning Simulation Models: A Case of Study and Results, *Computational Mathematical Organization Theory,* 1(2), pp. 123-141.

2. Bonacina, Maria Paola (1998), Theorem proving strategies: a search-oriented taxonomy (Position paper). In *Proceedings of the Second International Workshop on First-order Theorem Proving (FTP)*, Schloss Wilhelminenberg, Vienna, Austria, November 1998. Technical Report E1852-GS-981, Technische Universität Wien, 256-259.

3. Castelfranchi, C. and Y. Lesperance (Editors), *Intelligent Agents VII. Agent Theories, Architectures, and Languages. --- 7th International Workshop, ATAL-2000*, Boston, MA, USA, July 7--9, 2000, Proceedings, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag.

4. Chiang, C.L., and R. C.T. Lee (1973), *Symbolic Logic and Mechanical Theorem Proving*. London: Academic Press.

5. Domingo C., G. Tonella and O. Terán (1996), Generating Scenarios by Structural Simulation, in *AI, Simulation and Planning High Autonomy Systems,* pp 331-336. San Diego: The Univ. of Arizona.

6. Edmonds, B.(1999), Modelling Bounded Rationality In Agent-Based Simulations using the Evolution of Mental Models. In *Computational Techniques for Modelling Learning in Economics,* (Ed.) Brenner, T., 305-332. Kluwer.

7. Edmund, Ronald, Moshie Sipper and Matieu Capcarrere (1999), Design, Observation, Surprise! A test for emergence, *Artificial Life* (5), 225-239, Massachusetts Institute of Technology, Massachusetts, USA.

8. Engelfriet, J., C. Jonker and J. Treur (1998), Compositional Verification of Multi-Agent Systems in Temporal Multi-Epistemic Logic. Amsterdam, The Netherlands: AI Group, Vrije Universiteit.

9. Frühwirth, Thom (1994), Constraint Handling Rules, *Constraint Programming Basics and Trends*, Lectures Notes in Computer Science 910, Andreas Podelski Editor, Springer-Verlag, Berlin, 90-107.

10. Frühwirth, T., A. Herold, V. Küchenhoff, T. Le Provost, P. Lim, E. Monfroy and M. Wallace (1992). Constraint Logic Programming - An Informal Introduction, Chapter in *Logic Programming in Action*, (Eds.) G. Comyn et al., Springer LNCS 636, September.

11. Koen, V.Hindriks, Frank S. de Boer, Wiebe van der Hoek and John-Jules Ch. Meyer, Agent Programming with Declarative Goals, in (Castelfranchi et al., 2000).

12. Konolige, K. (1995), Autoepistemic Logic, in *Handbook of Logic in Artificial Intelligence and Logic Programming* (4), pp. 217-295. Oxford: Oxford Science Publications.

13. McCune, W. (1995), *OTTER 3.0 Reference Manual Guide*. Argonne, IL: Argonne National Laboratory.

14. Moss, S., H. Gaylard, S. Wallis, B. Edmonds (1998), SDML: A Multi-Agent Language for Organizational Modelling, *Computational Mathematical Organization Theory*, 4(1), 43-69.

15. Moss, S., B. Edmonds, S. Wallis (1997), Validation and Verification of Computational Models with Multiple Cognitive Agents, *Centre for Policy Modelling Report CPM-97-25*. Manchester: Centre for Policy Modelling (accessible at: http://www.cpm.mmu.ac.uk/cpmrep25.html)

16. Riley, Patrick, Peter Stone and Manuela Veloso, Layered Disclosure: Revealing Agents' Internals, in (Castelfranchi et al., 2000).

17. Wos, L. (1988), *Automated Reasoning: 33 Basis Research Problems*. New Jersey, USA: Prentice Hall.

18. Wack, P. (1985a), Scenarios: Uncharted Waters Ahead, *Harvard Business Revie*w, 63(5), September-October, pp. 73-89.

19. Wack, P. (1985b), Scenarios: Shootings The Rapids, *Harvard Business Review*, 63(6), November-December, pp. 139-150.

20. Zeigler, B. (1976), *Theory of Modelling and Simulation*. Malabar, Fl, USA: Robert E. Krieger Publishing Company.